JSON型Web APIのデータを表示するiOSアプリの作成

常三島技術部門
情報システムグループ　　宮武　秀考 (Hidetaka Miyatake)

１．はじめに

モバイル通信サービスは，情報入手・伝達手段の基幹となっており，モバイルデバイスからネットサービスを利用する人が増加している。今後も，モバイルデバイスに関する重要性が増加していくと予測できる。

そこで，モバイルアプリ開発の技術習得を目的にWeb APIからデータを取得し，表示を行うiOSアプリ(図１)を作成した。

当初，Androidアプリの作成も検討したが，2017年5月Google I/OでKotlinがAndroid開発言語として採用され，検討段階でKotlinに関する情報が少なかった。また，言語が統合開発環境Android Studioに正式対応していなかったので，iOSを選択した。

本稿では，このiOSアプリ作成について報告する。

２．開発環境

開発環境としてXcode8.3.3(Swift3.1)を用いた。Web APIからのデータ取得と処理に関して，MIT License[1]として公開されている，

・Alamofire4.5.1[2]: HTTP通信を扱うためのライブラリ
・SwiftyJSON3.1.4[3]: JSON形式をパースしSwiftで処理するためのライブラリ

を使用した。

iOS9から実装されたATSにより，自己署名証明書を使用したものやTLSv1.2未満など，セキュリティ要件を満たさないものは通信できない。今回，シミュレータによる確認までなので，ATSを無効にした。

iOSの開発において，UIコントローラの配置や画面遷移などを視覚的に設定できるStoryboardがある(図２)。画面レイアウトが分かりやすいが，UIコントローラのプロパティが分散され，どこで設定されているか分かりにくくなる。また，Gitでのソース管理・差分管理も難しくなる。

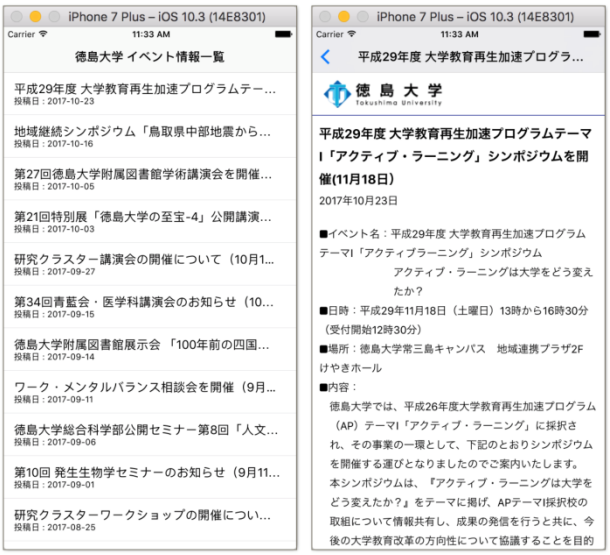

図１　Web APIのデータを表示するアプリ

本稿では，Storyboardを使用しないコードベースによるアプリ作成について説明する。

３．Web API

Web APIのレスポンスには，軽量なデータ記述が可能なJSON形式を用いることが多い。

徳島大学のイベント情報を表示するアプリ作成にあたり，大学サーバではJSON型Web APIを提供していないので，図３のようなJSON形式のファイルを作成し，別サーバで一時的に公開した。取得するファイルは静的なものであるが，作成したアプリは一般的なWeb APIで動作するようにした。

４．Web APIからのデータ取得

AlamofireでWeb APIからデータを取得するためにrequestメソッドを使用する。接続先を引数urlにString型で指定する。HTTPメソッドはデフォルトでGETが指定されているが，引数methodで変更が可能である。図４のようにAlamofire.request(url)で指定した接続先のWeb APIへリクエストを行う。

次にこのリクエストに対してresponseJSONメソッドを指定する。このメソッドの引数に

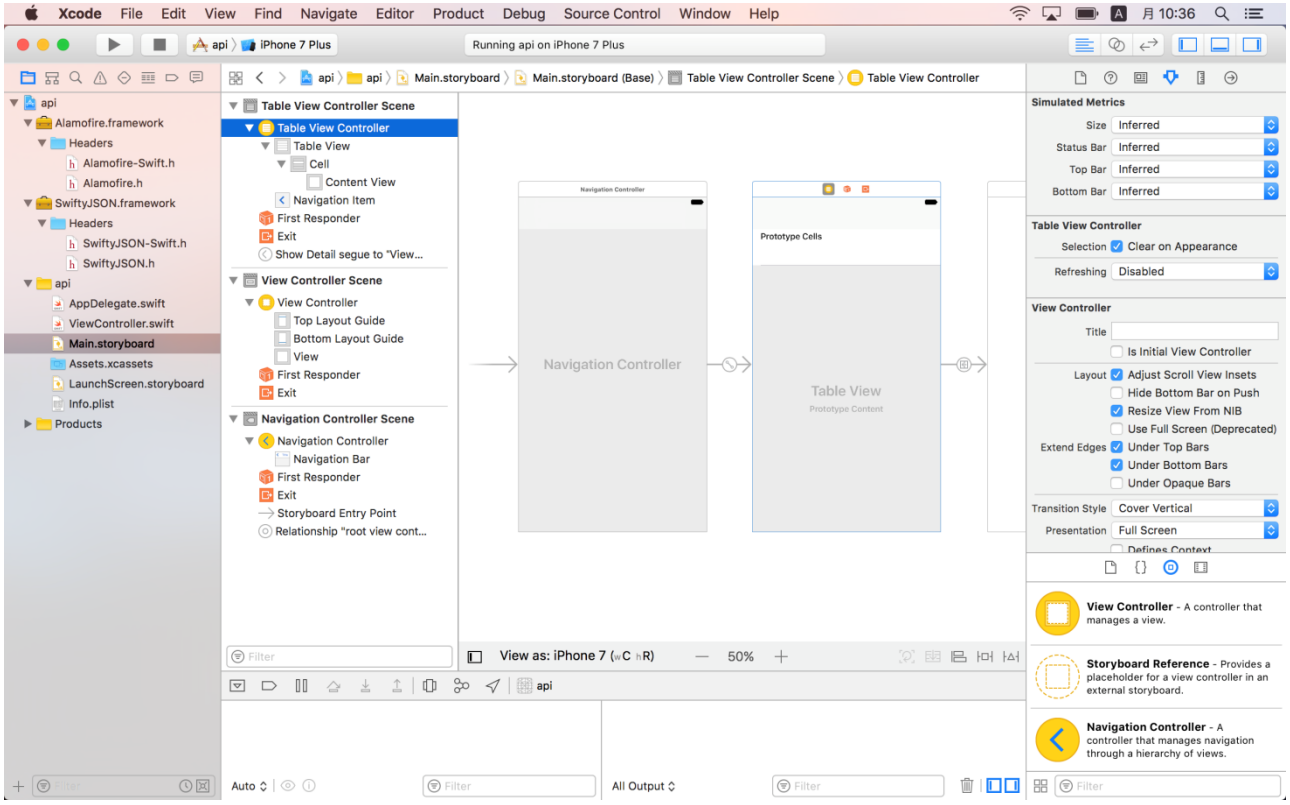

図２　Storyboardを用いた画面レイアウト

(Alamofire.DataResponse<Any>型)->Void型の関数を指定する。DataResponseはいくつかのプロパティを持つが，その1つに，

public let result: Alamofire.Result<value>

がある。Result(列挙体)はsuccessまたはfailureの値を取る。データが正常に取得できた場合，その結果をvalueプロパティで受け取り変数jsonへ代入する。これをクロージャで示すと図４になる。

変数jsonからデータを1件ごと配列itemsへ格納するためにforEachメソッドを使用する。forEachメソッドはSwift標準ライブラリで定義され，各要素に対して処理を繰り返す。JSON型にもforEachメソッドを使うことができ，引数には(String型，JSON型)->Void型の関数を指定する。これをクロージャで示すと図５になる。

以上により，Web APIから取得したデータをJSON型の配列itemsへ格納できる。

５．TableViewでの表示

Storyboardを使用した場合，UIコントローラを配置してテーブルを画面上へ表示させる。使用しない場合，図６のようにUITableViewクラスからインスタンスを生成し，必要なプロパティを設定後，addSubviewメソッドで画面上へ表示させる。

```
[{"title": "平成 29 年度　大学教育再生加速…",
 "date": "2017-10-23",
 "url": "http://www.tokushima-u.ac.jp/doc..."},
   … ]
```

図３　JSON形式のファイル

```
Alamofire.request(url)
  .responseJSON{response in
    var json:JSON
    json=JSON(response.result.value!)
}
```

図４　Web APIからデータを取得

配列itemsのデータをテーブルに表示するために，UITableViewクラスから生成したtableViewインスタンスを使用する。しかし，このクラスから生成したインスタンスではテーブルに値を表示できない。そこでdelegateとdataSourceプロパティへself(ViewController

クラスのインスタンス)を代入し，

```
tableView.delegate=self
tableView.dataSource=self
```

UITableViewクラスで処理できないテーブル処理をViewControllerクラスへ委譲させる。

また，デリゲート(委譲されたクラス)はプロトコルを批准しなければならないので，UITableViewクラスには，

```
var dataSource: UITableViewDataSource?
var delegate: UITableViewDelegate?
```

のようにオプショナル型のプロトコルが定義されている。それゆえ，selfは2つのプロトコルを批准していなければならず，ViewControllerクラスにこの2つのプロトコルを批准させる必要がある。批准すると，そこで定義されている2つのメソッド，

```
func tableView(_:numberOfRowsInSection:)
func tableView(_:cellForRowAt:)
```

を必ず実装しなければならない。

前者はテーブルのセル数をInt型で返すメソッド，後者はUITableViewCellクラスのオブジェクト(セルの内容)を返すメソッドである。tableView(_:cellForRowAt:)メソッドでセルデータからタイトルを取り出す場合，items[indexPath.row][“title”].stringとする。

以上により，配列itemsのデータをテーブルへ表示させることができる。

６．画面遷移

UITableViewクラスを利用してテーブルを表示させる画面(図１左)を作成した。遷移後の画面を図６と同様にUIWebViewクラスを利用して実装した。これは，指定したurlのWebページを表示する画面(図１右)である。

画面遷移はUINavigationControllerクラスのpushViewControllerメソッドで行う。

７．クラス間の変数参照

画面遷移で変数の受け渡しをする場合，AppDelegateクラスを用いる。URLの値を受け渡す場合，このクラスでurlプロパティを「var url: String?」と定義する。

遷移前と遷移後の両方のクラスに「let appDelegate = UIApplication.shared.delegate as! Appdelegate」と定義する。デーブルからタップされたURLの値を遷移前にプロパティurlへ代入し「appDelegate.url=“URL”」，遷移後は「appDelegate.url」で参照する。

以上により，タップした項目のURLが遷移後に渡され，Webページが表示される。

```
var items:[JSON]=[]
Alamofire.request(url)
 .responseJSON{response in
   var json:JSON
   json=JSON(response.result.value!)
   json.forEach{(_,data) in
     self.items.append(data)
   }
}
```

図５　図４にforEachメソッドの処理を追加

```
let tableView=UITableView()
tableView.frame=CGRect(x:y:width:height:)
self.view.addSubview(tableView)
```

図６　tableViewメソッドによる画面配置

８．まとめ

本稿では，JSON形式のファイルを受信し，その情報をUITableViewとUIWebViewを利用してiOSアプリを作成した。

iOSもAndroidも毎年秋頃にOSのアップデートがあり，その度にベータ版OSでアプリの動作確認が必要になる。アプリ作成のコストよりも，保守管理のコストの方が大きくなることもある。事前にモバイルアプリではなくWebアプリによるサービスで代用できるかを検討することが重要である。

参考文献

[1] The MIT License, https://opensource.org/licenses/mit-license.php

[2] Copyright(c) 2014-2017 Alamofire Software Foundation, http://alamofire.org/

[3] Copyright(c) 2017 Ruoyu Fu, https://github.com/SwiftyJSON/